

AD/ADVANTAGE

MANTIS for Windows Quick Reference

P19-2303-01

AD/Advantage® MANTIS for Windows Quick Reference

Publication Number P19-2303-01

© 1989, 1992, 1996, 1998, 1999 Cincom Systems, Inc.
All rights reserved

This document contains unpublished, confidential, and proprietary information of Cincom. No disclosure or use of any portion of the contents of these materials may be made without the express written consent of Cincom.

The following are trademarks, registered trademarks, or service marks of Cincom Systems, Inc.:

AD/Advantage®
AuroraDS®
CINCOM®
CINCOM SYSTEMS®



CONTROL™
CONTROL:Financial™
CONTROL:Manufacturing™
CPCS™

DOCVIEW™
EAGLE ADVANTAGE™
Enterprise Analyst Series™
MANTEXT®
MANTIS®
META*STAR®
M/Archive™
M/Exchange™
M/Graph™
M/Post™

M/Spell™
M/Text™
NORMAL®
PC CONTACT®
SPECTRA™
SUPRA®
SUPRA® Server
The Smart Choice®
TIS/XA™
TOTAL®
TOTAL FrameWork®

All other trademarks are trademarks or registered trademarks of:

Acucobol, Inc.
AT&T
Data General Corporation
Digital Equipment Corporation
Gupta Technologies, Inc.
International Business Machines Corporation

JSB Computer Systems Ltd.
Micro Focus, Inc.
Microsoft Corporation
Systems Center, Inc.
TechGnosis International, Inc.
UNIX System Laboratories, Inc.

or of their respective companies.

Cincom Systems, Inc.
55 Merchant Street
Cincinnati, OH 45246-3732
U.S.A.

PHONE: (513) 612-2300
FAX: (513) 612-2000
WORLD WIDE WEB: <http://www.cincom.com>

Attention:

Some Cincom products, programs, or services referred to in this publication may not be available in all countries in which Cincom does business. Additionally, some Cincom products, programs, or services may not be available for all operating systems or all product releases. Contact your Cincom representative to be certain the items are available to you.

Release information for this manual

AD/Advantage MANTIS for Windows Quick Reference, P19-2303-01, is dated November 30, 1998. This document supports Release 2.2.01 and higher of MANTIS for Windows.

We welcome your comments

We encourage critiques concerning the technical content and organization of this manual. A [Reader Comment Sheet](#) is included at the end of the manual for your convenience.

Cincom Technical Support for AD/Advantage

FAX: (513) 612-2000
Attn: MANTIS Support

E-mail: helpna@cincom.com

Phone: 1-800-727-3525

Mail: Cincom Systems, Inc.
Attn: MANTIS Support
55 Merchant Street
Cincinnati, OH 45246-3732
U.S.A.

Contents

About this book	vii
Using this document.....	vii
Document organization.....	vii
Conventions.....	viii
MANTIS documentation series.....	xi
Compatibility	13
Keys	17
Cursor control keys.....	17
Windowing keys.....	19
Action keys	20
Macro keys	21
Screen design keys	22
Editing commands	23
Statements and functions	33
Special characters	71

Attributes	73
Field-level attributes	74
Map-level attributes	75
Device-level attributes	76
Operators	77
Index	79

About this book

Using this document

MANTIS® is an application development system that consists of design facilities (e.g., screens and files) and a programming language. This manual provides a quick overview of MANTIS for Windows.

Document organization

The information in this manual is organized as follows:

Chapter 1—Compatibility

Discusses compatibility issues between MANTIS for Windows and MANTIS for Mainframe.

Chapter 2—Keys

Describes the key equivalents used with MANTIS for Windows.

Chapter 3—Editing commands

Provides brief descriptions of the editing commands.

Chapter 4—Statements and functions

Provides brief descriptions of the statements and functions.

Appendix A—Special characters

Provides brief descriptions of the special characters.

Appendix B—Attributes

Describes the field-level, map-level, and device-level attributes.

Appendix C—Operators

Describes the arithmetic operators.

Index

Conventions

The following table describes the conventions used in this document series:

Convention	Description	Example
Constant width type	Represents screen images and segments of code.	Screen Design Facility GET NAME LAST INSERT ADDRESS
Slashed b (<i>b</i>)	Indicates a space (blank). The example indicates that a password can have a trailing blank.	WRITEPASS <i>b</i>
Brackets []	Indicate optional selection of parameters. (Do not attempt to enter brackets or to stack parameters.) Brackets indicate one of the following situations: A single item enclosed by brackets indicates that the item is optional and can be omitted. The example indicates that you can optionally enter a program name.	COMPOSE [<i>program-name</i>]
	Stacked items enclosed by brackets represent optional alternatives, one of which can be selected. The example indicates that you can optionally enter NEXT, PRIOR, FIRST, or LAST. (NEXT is underlined to indicate that it is the default.)	<div><div>NEXT PRIOR FIRST LAST</div></div>

Convention	Description	Example
Braces { }	<p>Indicate selection of parameters. (Do not attempt to enter braces or to stack parameters.) Braces surrounding stacked items represent alternatives, one of which you must select.</p> <p>The example indicates that you must enter FIRST, LAST, or a value for <i>begin</i>.</p>	<pre>{ FIRST begin LAST }</pre>
<u>Underlining</u>	<p>Indicates the default value supplied when you omit a parameter.</p> <p>The example indicates that if you do not specify ON, OFF, or a row and column destination, the system defaults to ON.</p> <p>Underlining also indicates an allowable abbreviation or the shortest truncation allowed.</p> <p>The example indicates that you can enter either PRO or PROTECTED.</p>	<pre>SCROLL [ON OFF [row] [, col]]</pre> <pre><u>PRO</u>TECTED</pre>
Ellipsis points...	<p>Indicate that the preceding item can be repeated.</p> <p>The example indicates that you can enter (A), (A,B), (A,B,C), or some other argument in the same pattern.</p>	<pre>(argument , ...)</pre>
SMALL CAPS	Represent a keystroke. Multiple keystrokes are hyphenated.	ALT-TAB

Convention	Description	Example
UPPERCASE	Indicates MANTIS reserved words. You must enter them exactly as they appear. The example indicates that you must enter CONVERSE exactly as it appears.	CONVERSE name
lowercase	Indicates generic names of parameters for which you supply specific values as needed.	COMPOSE[program-name]
<i>Italics</i>	Indicate variables you replace with a value, a column name, a file name, and so on. The example indicates that you can supply a name for the program.	COMPOSE [<i>program-name</i>]
Punctuation marks	Indicate required syntax that you must code exactly as presented. () parentheses . , comma : colon ' ' single quotation marks	[LET] _p $\begin{bmatrix} (i) \\ (i,j) \end{bmatrix}$ [ROUNDED(<i>n</i>)] = <i>e1</i> [<i>e2, e3...</i>]

MANTIS documentation series

MANTIS is fourth-generation programming language used for application development. MANTIS is part of AD/Advantage[®], which offers additional tools for application development. Below are listed the manuals offered with MANTIS for Windows, organized by task. You may not have all the manuals that are listed here.

Getting started

- ◆ *MANTIS for Windows Administration Guide*, P19-2304*

General use

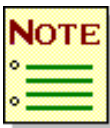
- ◆ *MANTIS for Windows Language Reference Manual*, P19-2302
- ◆ *MANTIS for Windows Facilities Reference Manual*, P19-2301
- ◆ *MANTIS for Windows Quick Reference*, P19-2303

SQL support

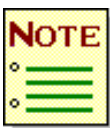
- ◆ *MANTIS for Windows SQL Support for SUPRA Programming Guide*, P19-2107
- ◆ *MANTIS for Windows SQL Support for SUPRA Administration Guide*, P19-2108*

Master user tasks

- ◆ *MANTIS for Windows Administration Guide*, P19-2304*
- ◆ *MANTIS for Windows SQL Support for SUPRA Administration Guide*, P19-2108*



Manuals marked with an asterisk (*) are listed twice because you use them for different tasks.



MANTIS educational material is available from your regional Cincom education department.

1

Compatibility

Size Limit	Personal computer ¹	IBM Mainframe
Maximum string length (bytes)	254–32750	254
Maximum dimension size (bytes)	255–16000	255
Maximum number of dimensions ²	2–255	2 (1 for TEXT)
Maximum program line number	9999–64000	30000
Maximum number of user words (variables)	2048–65535	2048
Maximum number of external DO levels	5–255	5
Maximum number of CHAIN parameters	40–255	40
Maximum program size (bytes)	32768–65248	32768

¹ Size limits on the personal computer must be in the ranges listed here.

² The string maximum length subscript is counted in an array dimension count. Therefore, if the maximum number of dimensions is 2, TEXT(15,4,16) is not allowed.

The following statements, functions, and commands are supported by MANTIS for the IBM mainframe, but not by MANTIS for Windows:

ASI	TOTAL
DEQUEUE	VIEW
ENQUEUE	VSI
MARK	

The following statements, functions, and commands are supported by MANTIS for Windows, but not by MANTIS for the IBM mainframe:

BREAK ¹	NEXT ¹
CHR ¹	NULL ¹
EDIT ^{1 2}	NUMERIC ¹
FOR ¹	RETURN ¹
GO ²	SET ^{1 2}
LANGUAGE ¹	\$SYMBOL ¹
	UPPERCASE ¹

¹ Documented in *MANTIS for Windows Language Reference Manual*, P19-2302.

² Documented in *MANTIS for Windows Facilities Reference Manual*, P19-2301.

The following statements and functions work differently on the personal computer and mainframe (refer to the *MANTIS for Windows Language Reference Manual*, P19-2302, for compatibility considerations for these statements):

ACCESS	KEY
ATTRIBUTE	OUTPUT
CALL	PERFORM
CHAIN	PRINTER
CLEAR	RELEASE
COMMIT	SCROLL
CONVERSE	SHOW
DO	SIZE
FSI	SLICE
GET	SLOT
INTERFACE	

The following facilities and editing commands work differently on Windows and mainframe (refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301, for compatibility considerations for these topics):

Screen Design	DOWN
Interface Design	QUIT
Transfer Facility	REPLACE
Windowing	RUN
External File Design	SAVE
Program Design	SEQUENCE
BIND	UP

2

Keys

Cursor control keys

Logical key	PC key	Enables you to . . .
UP	↑	Move the cursor up one row.
DOWN	↓	Move the cursor down one row.
LEFT	←	Move the cursor left one column.
RIGHT	→	Move the cursor right one column.
TAB	→ (TAB)	Move the cursor to the start of the next unprotected field.
BACKTAB	SHIFT-→ (SHIFT-TAB)	Move the cursor to the start of the current unprotected field (or to the start of the previous unprotected field if already at the start of the current field).
DELETE	DEL	Delete the character at the cursor.
BACKSPACE	← (BACKSPACE)	Delete the character to the left of the cursor and move the cursor one space to the left.
ERASEEOF	F6 or CTRL-END	Delete the character(s) from the cursor to the end of the field.
INSERT	INS	Turn on/off insert mode.
HOME	HOME	Move the cursor to the first unprotected field on the screen.

Logical key	PC key	Enables you to . . .
NEWLINE	CTRL-ENTER	Move cursor to the first unprotected field on the next line.
REFRESH	CTRL-R	Redisplay the current screen.
SELECT	CTRL-S	Copy a line from the scroll map to the Unsolicited Input field for you to modify.
SELUP	CTRL-U	Copy the previous input line from the scroll map to the Unsolicited Input field.
SELDOWN	CTRL-D	Copy the next input line from the scroll map to the Unsolicited Input field.
TABWRD	CTRL-→	Move the cursor to the next word.
TABBOW	CTRL-←	Move the cursor to the beginning of the word (or to the previous word if the cursor is currently at the beginning of a word).
TABEOF	END	Move cursor past last nonblank character in current field.
LINEDRAW	CTRL-B	Draw lines in the Screen Design Facility by using the cursor keys.
LINECLEAR	CTRL-N	Erase line-drawing characters in the Screen Design Facility by using the cursor keys.
STOP	CTRL-\	Terminate MANTIS. You must press this key twice, consecutively.

Windowing keys

Logical key	PC key	Enables you to. . .
WINUP	PGUP	Move the window up by the row increment value.
WINDOWN	PGDN	Move the window down by the row increment value.
WINLEFT	CTRL-PGUP	Move the window left by the column increment value.
WINRIGHT	CTRL-PGDN	Move the window right by the column increment value.
WINTOPL	CTRL-HOME	Move the window to the top left of the logical display.
WINTOPR	No default (user definable)	Move the window to the top right of the logical display.
WINBOTL	No default (user definable)	Move the window to the bottom left of the logical display.
WINBOTR	No default (user definable)	Move the window to the bottom right of the logical display.
SCROLLALL	CTRL-A	Display the entire scroll output map by scrolling from top to bottom.
INPUTMAP	CTRL-I	Add or remove the Unsolicited Input field and Key Simulation field.
WINDOWMAP	CTRL-W	Add or remove the row/column coordinates at the bottom of the screen.
STATUSLINE	CTRL-P	Add or remove the status line on the last line of the screen. The status line shows the insert mode and cursor position.
VALIDINFO	CTRL-V	Display extended edit attributes for a particular field. Position cursor over the field and press VALIDINFO any time during data entry to the screen.

Action keys

The function of each of the following keys is determined by the application:

Logical key	PC key	Logical key	PC key
CANCEL	ESC	PF11	ALT--
CLEAR*	F2	PF12	ALT-+
ENTER	ENTER	PF13	ALT-Q
PA1	ALT-J	PF14	ALT-W
PA2	ALT-K	PF15	ALT-E
PA3	ALT-L	PF16	ALT-R
PF1	ALT-1	PF17	ALT-T
PF2	ALT-2	PF18	ALT-Y
PF3	ALT-3	PF19	ALT-U
PF4	ALT-4	PF20	ALT-I
PF5	ALT-5	PF21	ALT-O
PF6	ALT-6	PF22	ALT-P
PF7	ALT-7	PF23	ALT-A
PF8	ALT-8	PF24	ALT-S
PF9	ALT-9		
PF10	ALT-0		

* This key erases the entire screen.

Macro keys

Logical key	PC key	Enables you to
EDIT*	CTRL-E	Invoke the user-specified editor.
HELP*	CTRL-H	Display help prompter.
KILL	CTRL-K	End program loop.
QUIT*	CTRL-Q	Exit from programming mode and return to the Program Design Facility menu.

- * The function of this key is application-dependent. The description here is the intended purpose of the key. Consult each facility for specific details.

Screen design keys

The following list describes how the screen design keys work:

Key	Function
PF1	Inserts line before the field identified by the cursor position if at least one empty line is available in the specified screen domain.
PF2	Deletes line to right of cursor.
PF3	Removes, displays, or moves column scale line at cursor.
PF4	Moves field at cursor.
PF5	Copies field at cursor.
PF6	Deletes field at cursor.
PF7	Inserts specified number of lines.
PF8	Deletes specified number of lines.
PF9	Removes or adds the row scale.
PF10	Moves range of lines.
PF11	Copies range of lines.
PF12	Displays screen domain size on last line of screen.
PF14	Invokes your text editor.
HELP ¹	Displays a help screen.
EDIT ¹	Invokes your text editor.
LINEDRAW ²	Enables you to draw lines in the Screen Design Facility using cursor keys.
LINECLEAR ²	Enables you to erase line-drawing characters in the Screen Design Facility using cursor keys.

¹ For more information about specific key combinations, see “Macro keys” on page 21.

² For more information about specific key combinations, see the table under “Cursor control keys” which starts on page 17.

3

Editing commands

ALTER command

The ALTER command allows you to modify lines of a MANTIS program without retyping the lines. The following example shows the syntax for this command:

ALTER *begin*[,*end*]

Example

```
ALTER 40,60
```

BIND command

The BIND command converts a program from unbound to bound or from bound to unbound format. The following example shows the syntax for this command:

BIND

ON
OFF

Example

```
BIND
```

CHANGE command

The CHANGE command replaces occurrences of a specified string with a new string over a range of lines. The following example shows the syntax for this command:

CHANGE [**ALL**] $\left\{ \begin{array}{c} ' \\ , \\ : \\ ; \end{array} \right\}$ *old-string* $\left\{ \begin{array}{c} ' \\ , \\ : \\ ; \end{array} \right\}$ *new-string* $\left\{ \begin{array}{c} ' \\ , \\ : \\ ; \end{array} \right\}$

Example

```
CHANGE ALL 'J'K'40
```

CLEAR command

The CLEAR command clears program breakpoints. The following example shows the syntax for this command:

CLEAR BREAK [*line-number*]

Example

```
CLEAR BREAK 60
```

COPY command

The COPY command copies portions of a program into the work area. The following example shows the syntax for this command:

COPY [*libname,*] $\left\{ \begin{array}{c} \text{FIRST} \\ \text{begin} \\ \text{LAST} \end{array} \right\} \left[\begin{array}{c} , \text{end} \\ , \text{LAST} \end{array} \right] \left\{ \begin{array}{c} \text{FIRST} \\ \text{AFTER position} \\ \text{LAST} \end{array} \right\}$

Example

```
COPY SIMPLE_INTEREST,70,80 AFTER 34
```

DISPLAY command

The DISPLAY command displays attributes (e.g., type and value) of specified MANTIS variables. The following example shows the syntax for this command:

```
DISPLAY { name
          ALL } ,...
```

Example

```
DISPLAY I,J,K
```

DOWN command

The DOWN command selects as the current program an external subroutine at a lower level. The following example shows the syntax for this command:

```
DOWN [levels]
```

Example

```
DOWN 2
```

EDIT command

The EDIT command invokes an external text editor. The following example shows the syntax for this command:

```
EDIT [libname [, password]]
     [FILE file - spec]
```

Example

```
EDIT EXAMPLE_PROGRAM
```

ERASE command

The ERASE command deletes one or more lines of a MANTIS program. The following example shows the syntax for this command:

ERASE *begin[,end]*

Example

```
ERASE 70,80
```

GO command

The GO command resumes execution of a program after the program stops at a breakpoint. The following example shows the syntax for this command:

GO [*lines*]

Example

```
GO 1
```

HELP command

The HELP command provides information on an error message, command, or statement; or displays a list of reserved words, commands, statements, and functions for which help is available. The following example shows the syntax for this command:

HELP	[<i>message – number</i>]
		<i>keyword</i>	
		RESERVED	
		LAST	

Example

```
HELP CONVERSE
```

LIST command

The LIST command lists all or part of the program currently in the work area. The following example shows the syntax for this command:

LIST [*begin*[,*lines*]]

Example

LIST

LOAD command

The LOAD command retrieves a program from a MANTIS library or a PC file and places it in your work area. The following example shows the syntax for this command:

LOAD { *libname*[,*password*] }
 { **FILE** *file* – *spec* }

Example

LOAD EXAMPLE_PROGRAM

NEW command

The NEW command clears the work area. The following example shows the syntax for this command:

NEW

Example

NEW

PURGE command

The PURGE command deletes a program from your library but not from the work area. The following example shows the syntax for this command:

PURGE *libname*[,*password*]

Example

```
PURGE EXAMPLE_PROGRAM
```

QUIT command

The QUIT command terminates programming mode and returns you to the Program Design Facility menu. The following example shows the syntax for this command:

QUIT

Example

```
QUIT
```

REPLACE command

The REPLACE command replaces a program in your library or in a PC file with the program currently in your work area. The following example shows the syntax for this command:

REPLACE $\left\{ \begin{array}{l} \left[\textit{libname} \right] \left[\textit{password}, \textit{description} \right] \\ \textit{FILE} \left[\textit{file} - \textit{spec} \right] \end{array} \right\}$

Example

```
REPLACE PROGRAMA, ,TEST PROGRAM
```

RUN command

The RUN command executes the program in the work area, optionally loading it into the work area from a specified library. The following example shows the syntax for this command:

```
RUN [libname] [begin]
```

Example

```
RUN
```

SAVE command

The SAVE command copies the program in the work area into your library or a PC file. The following example shows the syntax for this command:

```
SAVE { [libname] [password [, description]]  
        FILE [file – spec] }
```

Example

```
SAVE PROGRAMA, , TEST PROGRAM
```

SEQUENCE command

The SEQUENCE command renumbers lines of the program in the work area. The following example shows the syntax for this command:

```
SEQUENCE [first [, increment]] [entry-name]
```

Example

```
SEQUENCE
```

SET command

The SET command sets breakpoints in the current program. The following example shows the syntax for this command:

SET BREAK *line-number*[,*command*]

Example

```
SET BREAK 70, "SHOW I"
```

SHOW command

The SHOW command shows either program breakpoints or the next line for the program to execute. The following example shows the syntax for this command:

SHOW { **BREAK**
 NEXT }

Example

```
SHOW BREAK
```

UP command

The UP command selects as the current program a program or an external subroutine at a higher level. The following example shows the syntax for this command:

UP [*levels*]

Example

```
UP 2
```

USAGE command

The USAGE command shows where a MANTIS reserved word, symbolic name, or string is used in your program. The following example shows the syntax for this command:

$$\text{USAGE} \left\{ \begin{array}{l} \textit{keyword} \\ \textit{symbolic} - \textit{name} \\ \textit{'string'} \end{array} \right\} [,\textit{begin}[, \textit{line}]]$$

Example

```
USAGE REC1
```


4

Statements and functions

ABS function

The ABS function returns the absolute value of an arithmetic expression. The following example shows the syntax for this function:

ABS(*a*)

Example

`ABS(-14E9)` returns 14E9

ACCESS statement

The ACCESS statement defines an external file view for program access. The following example shows the syntax for this statement:

ACCESS *access-name*(*libname,password*[,PREFIX**][,*levels*]**

[*,NEW*
,REPLACE*]** [**FILE *extname]),...

Example

`ACCESS RECORD("INDEX","SERENDIPITY",16)`

***access-name* function**

The *access-name* function returns a text value indicating the status of the most recent GET, UPDATE, INSERT, or DELETE operation on an external file view with the symbolic name *access-name*, as defined in an ACCESS statement. The following example shows the syntax for this function:

access-name

Example

```
IF RECORD="FOUND"
```

ATN function

The ATN function returns the angle in radians whose tangent is an arithmetic expression. The following example shows the syntax for this function:

ATN(*a*)

Example

```
ATN(100) returns 1.56079666011
```

ATTRIBUTE statement

The ATTRIBUTE statement changes attributes of a field, screen, or printer. The following example shows the syntax for this statement:

ATTRIBUTE $\left\{ \begin{array}{l} (\textit{screen - name} [, \textit{field - name}]) \\ (\textit{PRINTER}) \end{array} \right\} = \textit{attributes}, \dots$

Example

```
ATTRIBUTE ( INVOICE , ACCT_NUM ) = " CUR , BRI "
```

ATTRIBUTE function

The ATTRIBUTE function returns the attributes of the printer. The following example shows the syntax for this function:

ATTRIBUTE(PRINTER)

Example

```
SHOW ATTRIBUTE (PRINTER)
```

BIG statement

The BIG statement defines numeric variables or arrays of numeric variables which hold values with a maximum of 15 significant digits. The following example shows the syntax for this statement:

BIG *big-name*[(*dimension*,...)],...

Example

```
BIG ALPHA ( 64 , 3 ) , BETA
```

BREAK statement

The BREAK statement exits from a FOR-END, UNTIL-END, WHEN-END, or WHILE-END statement. The following example shows the syntax for this statement:

BREAK

Example

```
BREAK
```

CALL statement

The CALL statement calls an interface program (not supported under DOS). The following example shows the syntax for this statement:

CALL *interface-name*[(*argument*,...)]*[LEVEL=level-number]*

Example

```
CALL MASTER ( "GET" ,1234) LEVEL=2
```

CHAIN statement

The CHAIN statement replaces the program currently in the work area or at the current DOLEVEL with another MANTIS program and begins to execute that program. The following example shows the syntax for this statement:

CHAIN *libname*[,*argument*,...] *[LEVEL]*

Example

```
CHAIN "MASTER:PROGRAM"
```

CHR function

The CHR function returns a text value of one or more characters corresponding to the ASCII code(s) specified. The following example shows the syntax for this function:

CHR(*a*,...)

Example

```
CHR(97,98,99) returns "abc"
```

CLEAR statement

The CLEAR statement clears the screen, data referenced by a symbolic name, all program data, or the fault trap. The following example shows the syntax for this statement:

```
CLEAR 

|             |
|-------------|
| <i>name</i> |
| ALL         |
| TRAP        |

 ,...
```

Example

```
CLEAR COUNT, SUBTOTAL, CUSTFILE, CUSTSCREEN
```

COMMIT statement

The COMMIT statement indicates the completion of a Logical Unit of Work (LUW), or enables or disables automatic COMMIT processing. The following example shows the syntax for this statement:

```
COMMIT 

|     |
|-----|
| ON  |
| OFF |


```

Example

```
COMMIT
```

CONVERSE statement

The CONVERSE statement displays a map set of one or more screens, adds a screen map to the map set without displaying it, or removes a map from the map set. The following example shows the syntax for this statement:

	$[(row1, col1)] \left[\begin{array}{c} WAIT \\ SET \\ UPDATE \end{array} \right] \left[\left\{ \begin{array}{c} WINDOW \\ DISPLAY \end{array} \right\} \right]$
CONVERSE <i>screen - name</i>	$[(row2, col2)] [template]$ <p>RELEASE</p>

Example

```
CONVERSE MAP(10,45)
```

COS function

The COS function returns the cosine of an arithmetic expression in radians. The following example shows the syntax for this function:

COS(*a*)

Example

`COS(10)` returns `-.839071529`

CURSOR function

The CURSOR function determines whether the cursor was in the specified field after the last terminal input. The following example shows the syntax for this function:

CURSOR(*screen-name,field-name*)

Example

```
WHEN CURSOR (MENU_SCREEN,CLIENT_UPDATE)
```

DATAFREE function

The DATAFREE function returns the number of bytes currently available in your data area. The following example shows the syntax for this function:

DATAFREE

Example

```
SHOW DATAFREE
```

DATE function

The DATE function returns the text value of the current date in *YYYY/MM/DD* format. The following example shows the syntax for this function:

DATE

Example

```
LAST_UPDATED=DATE
```

DATE statement

The DATE statement specifies the date mask to be used by the DATE function. The following example shows the syntax for this statement:

DATE=*mask-expression*

DELETE statement

The DELETE statement deletes a record from a MANTIS file or from an external file. The following example shows the syntax for this statement:

DELETE *file-name* [LEVEL=*level-number*]

Example

```
DELETE RECORD
```

DEQUEUE statement

The DEQUEUE statement, provided for compatibility with MANTIS for the IBM mainframe, has no effect on MANTIS for Windows. The following example shows the syntax for this statement:

DEQUEUE *resource*

Example

```
DEQUEUE "CUSTOMERS"+RECORD_KEY
```

DO statement

The DO statement executes an internal or external subroutine. The following example shows the syntax for this statement:

DO $\left\{ \begin{array}{l} \textit{entry - name} \\ \textit{program - name} \end{array} \right\} [(\textit{argument}, \dots)]$

Example

```
DO EDIT RTN(TYPE,CUST_NO,STATUS,MESSAGE)
```

DOLEVEL function

The DOLEVEL function returns the current level in an external subroutine. The following example shows the syntax for this function:

DOLEVEL

Example

```
SHOW DOLEVEL
```

E function

The E function returns the value of natural logarithm base e (2.71828182846). The following example shows the syntax for this function:

E

Example

```
X=2*E+1
```

EDIT statement

The EDIT statement edits a text variable or array. The following example shows the syntax for this statement:

EDIT LIST *text-name*

Example

```
EDIT LIST SURNAMES
```

ENQUEUE statement

The ENQUEUE statement, provided for compatibility with MANTIS for the IBM mainframe, has no effect on MANTIS for Windows. The following example shows the syntax for this statement:

ENQUEUE *resource*

Example

```
ENQUEUE "CUSTOMERS"+RECORD_KEY
```

ENTRY-EXIT statements

The ENTRY-EXIT statements define the beginning and end of a subroutine or a program. The following example shows the syntax for this statement:

ENTRY *entry-name*[(*parameter*,...)]

.

statements

.

EXIT

Example

```
ENTRY DATA_ENTRY
```

.

.

.

```
EXIT
```

EXEC_SQL-END statements

The EXEC_SQL-END statements designate a block of embedded SQL statements. The following example shows the syntax for these statements:

```
EXEC_SQL[(exp1[exp2])][;SQL statement]
|
|
SQL statement (continued)...
|
|
END
```

FOR-END statements

The FOR-END statements execute a block of statements repeatedly while a counter is incremented or decremented through a specified range of values. The following example shows the syntax for these statements:

```
FOR counter = initial TO final [BY increment]
.
statements
.
END
```

Example

```
FOR I=1 TO SIZE(ARRAY,1)
.SHOW ARRAY(I)
END
```

FORMAT function

The FORMAT function returns the text value that results from converting a number using an edit mask. The following example shows the syntax for this function:

FORMAT(*number*, *mask*)

Example

```
FORMAT (46.35,"$*****.##") returns $****46.35
```

FSI function

The FSI function indicates the success or failure of an external file access or of an interface CALL. The following example shows the syntax for this function:

FSI(*name*[,*msg*])

Example

```
IF FSI(X,MESSAGE) = "UNAVAILABLE"  
.  
.  
.  
SHOW "TEST FAILED: STATUS="+MESSAGE
```

GET statement

The GET statement reads a record from a MANTIS file or an external file. The following example shows the syntax for this statement:

	<div style="display: inline-block; vertical-align: middle;"><div style="display: inline-block; vertical-align: middle;"><div style="display: inline-block; vertical-align: middle;">(key,...)</div><div style="display: inline-block; vertical-align: middle;"><div style="display: inline-block; vertical-align: middle;">NEXT</div><div style="display: inline-block; vertical-align: middle;">EQUAL</div></div></div></div>	
GET <i>file - name</i>	<div style="display: inline-block; vertical-align: middle;"><div style="display: inline-block; vertical-align: middle;">FIRST</div><div style="display: inline-block; vertical-align: middle;"><div style="display: inline-block; vertical-align: middle;">NEXT</div></div></div>	<div style="display: inline-block; vertical-align: middle;">[ENQUEUE][LEVEL = <i>level - number</i>]</div>

Example

```
GET RECORD("WILLIAMS") LEVEL=LEVEL_NUMBER
```

HEAD statement

The HEAD statement centers a heading on the top line of the screen in scroll mode. The following example shows the syntax for this statement:

HEAD heading

Example

```
HEAD "BUZZ PHRASE GENERATOR"
```

IF-ELSE-END statements

The IF-ELSE-END statements execute one block of statements if a specified condition is TRUE, another block if the condition is FALSE. The following example shows the syntax for these statements:

```
IF expression.  
[true-block]  
.  
ELSE  
.  
[false-block]  
.  
END
```

Example

```
IF A>B  
  .A=A+1  
ELSE  
  .B=B*5  
END
```

INSERT statement

The INSERT statement inserts a new record into a MANTIS file or an external file. The following example shows the syntax for this statement:

INSERT *file-name* [LEVEL=*level-number*]

Example

```
INSERT RECORD
```

INT function

The INT function returns the integer value of an arithmetic expression, truncating fractions. The following example shows the syntax for this function:

INT(*a*)

Example

```
INT(45.5) returns 45
```

INTERFACE statement

The INTERFACE statement defines an interface (not supported under DOS) for your program to access. The following example shows the syntax for this statement:

INTERFACE *interface-name* (*libname,password*[,*PREFIX*][,*levels*]),...

Example

```
INTERFACE MASTER( "CUSTOMERS" , "ALIBABA" , 10 )
```

***interface-name* function**

The *interface-name* function returns the 8-character interface status of the last CALL to *interface-name*. The following example shows the syntax for this function:

interface-name

Example

```
IF MASTER="ERROR"
```

KEY function

The KEY function returns a text value identifying your response to the most recent CONVERSE, OBTAIN, PROMPT, or WAIT statement. The following example shows the syntax for this function:

KEY

Example

```
UNTIL KEY="CANCEL"
```

LANGUAGE function

The LANGUAGE function returns the default language of the user currently signed on. The following example shows the syntax for this function:

LANGUAGE

Example

```
SHOW LANGUAGE
```

LET statement

The LET statement assigns the value of an expression to a variable, array element, or text substring, or assigns values to a sequence of consecutive array elements. The following example shows the syntax for this statement:

[LET] *variable* [ROUNDED[(*places*)]]=*expression*,...

Example

```
LET X=VALUE(Y)
```

LOG function

The LOG function returns the natural logarithm of an arithmetic expression (inverse of the EXP function). The following example shows the syntax for this function:

LOG(*a*)

Example

```
LOG(1000) returns 6.90775527898
```

MODIFIED function

The MODIFIED function determines how many fields were modified or whether a specific field on a screen was modified during the last CONVERSE statement. The following example shows the syntax for this function:

MODIFIED(*screen-name*[,*field-name*])

Example

```
IF MODIFIED(CLIENT_INFO)
```

NEXT statement

The NEXT statement transfers control to the next conditional repeat in a FOR-END, UNTIL-END, or WHILE-END statement, or to the next WHEN condition in a WHEN-END statement. The following example shows the syntax for this statement:

NEXT

Example

```
NEXT
```

NOT function

The NOT function returns TRUE if an arithmetic expression evaluates to FALSE; otherwise, NOT returns FALSE. The following example shows the syntax for this function:

NOT(*a*)

Example

```
IF NOT(A=3 OR J=TRUE)
```

NULL function

The NULL function returns an empty (zero-length) text value. The following example shows the syntax for this function:

NULL

Example

```
IF CLIENT_NAME=NULL
```

NUMERIC function

The NUMERIC function determines if a text expression contains a valid number. The following example shows the syntax for this function:

NUMERIC(*text-expression*)

Example

```
NUMERIC("123,456.789") returns TRUE
```

OBTAIN statement

The OBTAIN statement gets data values from an unformatted screen and assigns them to numeric or text variables, or array elements. The following example shows the syntax for this statement:

OBTAIN *variable*,...

Example

```
OBTAIN MONTH, DAY, YEAR
```

OUTPUT statement

The OUTPUT statement routes output from the CONVERSE and SHOW statements to the screen and/or printer. The following example shows the syntax for this statement:

OUTPUT	{	SCREEN	}
		PRINTER[VIA exit]	
		SCREEN PRINTER [VIA exit]	

Example

```
OUTPUT SCREEN PRINTER
```

PAD statement

The PAD statement inserts padding characters into a text variable, array element, or substring. The following example shows the syntax for this statement:

PAD <i>string</i> [<i>padding</i>]	BEFORE
	AFTER
	ALL

Example

```
PAD A "*" ALL results in A+***CLIENT NAME***
```

PASSWORD function

The PASSWORD function returns a text value containing the password entered during MANTIS sign-on. The following example shows the syntax for this function:

PASSWORD

Example

```
FILE REC("CUST FILE",PASSWORD)
```

PERFORM statement

The PERFORM statement invokes a DOS or OS/2 command. The following example shows the syntax for this statement:

PERFORM *command*

```
PERFORM "CHKDSK"
```

PI function

The PI function returns the value of pi (3.14159265359). The following example shows the syntax for this function:

PI

Example

```
X=100*PI
```

POINT function

The POINT function returns the character position where joining or removal occurs in a text expression. The following example shows the syntax for this function:

POINT(*tt*)**

Example

```
DECIMAL=POINT(AMOUNT-" . " )
```

PRINTER function

The PRINTER function returns the current printer assignment. The following example shows the syntax for this function:

PRINTER

Example

```
SHOW PRINTER
```

PRINTER= statement

This PRINTER= statement directs printer output to a file or device. The following example shows the syntax for this statement:

PRINTER=*printer-spec*

Example

```
PRINTER= "MANTIS.LST;PRINT #"
```

PROGFREE function

The PROGFREE function returns the number of bytes available in the program work area. The following example shows the syntax for this function:

PROGFREE

Example

```
SHOW PROGFREE
```

PROGRAM statement

The PROGRAM statement defines an external subroutine to be invoked by a DO statement. The following example shows the syntax for this statement:

PROGRAM *program-name*(*libname,password*),...

Example

```
PROGRAM EDIT_RTN( "VALIDATION" , "COMMON" )
```

PROMPT statement

The PROMPT statement displays a specified prompt. With chained prompts, MANTIS displays each prompt in the chain. The following example shows the syntax for this statement:

PROMPT *libname*

Example

```
PROMPT "MASTER:FACILITY HELP"
```

RELEASE statement

The RELEASE statement closes an external file, releases internal storage used by an external subroutine, or closes an interface. The following example shows the syntax for this statement:

RELEASE $\left\{ \begin{array}{l} \textit{access - name} \\ \textit{interface - name} \\ \textit{program - name} \end{array} \right\}$

```
RELEASE EDIT_RTN
```

RELEASE function

The RELEASE function gets the current release and version number of MANTIS and the operating environment under which MANTIS is running. The following example shows the syntax for this function:

RELEASE

Example

```
SHOW RELEASE
```

RESET statement

The RESET statement backs out a Logical Unit of Work (LUW). The following example shows the syntax for this statement:

RESET

Example

```
RESET
```

RETURN statement

The RETURN statement returns control from a subroutine or stops execution of a program. The following example shows the syntax for this statement:

RETURN

Example

```
RETURN
```

RND function

The RND function returns a random real number between zero and a specified number, excluding zero and that number. The following example shows the syntax for this statement:

RND(*a*)

Example

```
A=INT(RND(10)+1)
```

SCREEN statement

The SCREEN statement defines a screen design for use in a program. The following example shows the syntax for this statement:

SCREEN *screen-name*(*libname*[,**PREFIX**]),...

Example

```
SCREEN MAP ( " INDEX " )
```

screen-name function

The *screen-name* function returns a text value identifying the last key pressed in response to the most recent CONVERSE statement for this screen. The following example shows the syntax for this function:

screen-name

Example

```
WHILE MAP<>"CANCEL"
```

SCROLL statement

The SCROLL statement specifies the amount by which windowing keys will move the window. The following example shows the syntax for this statement:

SCROLL

ON
OFF
[<i>row1</i>][<i>col1</i>]

Example

```
SCROLL 11,40
```

SEED statement

The SEED statement causes the random number generator to create a new sequence of random numbers. The following example shows the syntax for this statement:

SEED

Example

SEED

SET statement

The SET statement sets a fault trap or assigns a value to a system environment variable. The following example shows the syntax for this statement:

$$\text{SET} \left\{ \begin{array}{l} \text{TRAP} \left\{ \begin{array}{l} \textit{entry - name} \\ \textit{program - name} \end{array} \right\} \\ \$\text{SYMBOL} \textit{name TO value} \end{array} \right\}$$

Example

SET TRAP FAULT

SGN function

The SGN function returns -1 if an arithmetic expression is less than 0, 0 if it equals 0, and +1 if it is greater than 0. The following example shows the syntax for this function:

SGN(*a*)

Example

SGN(-14) returns -1

SHOW statement

The SHOW statement displays data on the screen in scroll mode. The following example shows the syntax for this statement:

SHOW	[<i>data - item</i>]	...
		,		
		;		
		<i>AT column</i>		
		TRAP		

Example

```
SHOW AT 20 NUMBER,FIRST_NAME;LAST_NAME
```

SIN function

The SIN function returns the sine of an arithmetic expression in radians. The following example shows the syntax for this function:

SIN(*a*)

Example

```
SIN(100) returns      -.5063656411
```

SIZE function

The SIZE function returns the size and dimensions of an expression, a variable, or an array. The following example shows the syntax for this function:

SIZE	{	<i>text - expression</i>	[, "BYT"]
		<i>text - variable,</i>	
		<i>text - array</i>	, "MAX"
		<i>text - variable</i>	
		<i>arithmetic - variable</i>	, "DIM"
		<i>array</i>	
		<i>array, dimension</i>	

Example

```
SIZE(CUST_NAME, "MAX")
```

SLICE statement

The SLICE statement specifies the number of statements in a program slice. With SLOT, this determines how many statements MANTIS will execute before stopping with the message: POTENTIAL PROGRAM LOOP ENCOUNTERED. The following example shows the syntax for this statement:

SLICE statements

Example

```
SLICE 3000
```

SLOT statement

The SLOT statement specifies the number of program slices for MANTIS to execute before stopping with the message: POTENTIAL PROGRAM LOOP ENCOUNTERED. The following example shows the syntax for this statement:

SLOT *slices*

Example

```
SLOT 1
```

SMALL statement

The SMALL statement defines numeric variables or arrays of numeric variables, allocating space in the work area to hold their values with up to six significant digits. The following example shows the syntax for this statement:

SMALL *small-name*[(*dimension*,...)],...

Example

```
SMALL ALPHA(64,3),BETA(X)
```

SQLCA statement

The SQLCA statement specifies a value for your SQLCA element. The following example shows the syntax for this statement:

SQLCA(*element_name*)=*expression*

Example

```
SQLCA("DBTYPE")="SUPRA"
```

SQLCA function

The SQLCA function specifies the variable to receive the value of your SQLCA element. The following example shows the syntax for this function:

SQLCA(*element_name*)

Example

```
IF SQLCA( "SQLCODE" ) < 0
```

SQLDA statement

The SQLDA statement allocates/deallocates a new SQL Descriptor Area or moves data from your program into an SQL Descriptor Area. The following example shows the syntax for this statement:

SQLDA(*sqlda - name*) = $\left\{ \begin{array}{l} \text{NEW} \\ \text{QUIT} \end{array} \right\}$

SQLDA(*sqlda - name* $\left\{ \begin{array}{l} , \text{header_element} \\ , \text{repeating_element, index} \end{array} \right\}$) = *expression*

Example

```
SQLDA( "DA1 " , "SQLN" ) = 5
```

SQLDA function

The SQLDA function moves data from an SQL Descriptor Area into your program. The following example shows the syntax for this function:

SQLDA(*sqlda - name* $\left\{ \begin{array}{l} , \text{header_element} \\ , \text{repeating_element, index} \end{array} \right\}$)

Example

```
EMPLOYEE_NUMBER=SQLDA( "SQLDA1 " , "SQLDATA" , 1 )
```

SQR function

The SQR function returns the square root of an arithmetic expression. The following example shows the syntax for this function:

SQR(*a*)

Example

```
SQR(100) returns      10
```

STOP statement

The STOP statement terminates program execution and then returns you to your facility program unless you are in programming mode. The following example shows the syntax for this statement:

STOP

Example

```
STOP
```

\$SYMBOL function

The \$SYMBOL function returns the text value assigned to the DOS or OS/2 environment variable with a specified name. The following example shows the syntax for this function:

\$SYMBOL(*t*)

Example

```
SHOW $SYMBOL( "MONTH" )
```

TAN function

The TAN function returns the tangent of an arithmetic expression in radians. The following example shows the syntax for this function:

TAN(*a*)

Example

```
Y=TAN ( X+1 )
```

TERMINAL function

The TERMINAL function is provided for compatibility with MANTIS for the IBM mainframe. This function contains the name of the computer system on which MANTIS is running, as found in the Windows registry. The following example shows the syntax for this function:

TERMINAL

Example

```
SHOW TERMINAL
```

TERMSIZE function

The TERMSIZE function returns a text value giving the size of the screen in rows and columns in the format *rrXcc*. The following example shows the syntax for this function:

TERMSIZE

Example

```
SHOW TERMSIZE
```

TEXT statement

The TEXT statement defines text variables or arrays of text variables. The following example shows the syntax for this statement:

TEXT *text-name*[[(*dimension*,...,)*length*]],...

Example

```
TEXT ALPHA( 64, 3 )
```

TIME function

The TIME function returns a text value containing the current time in *HH:MM:SS* format. The following example shows the syntax for this function:

TIME

Example

```
SHOW TIME
```

TIME statement

The TIME statement specifies the time mask to be used by the TIME function. The following example shows the syntax for this statement:

TIME=*mask-expression*

Example

```
TIME="HH:MM:SS am"
```

TRAP statement

The TRAP statement determines whether error conditions in GET, UPDATE, INSERT, DELETE, and CALL processing will terminate program execution or whether MANTIS will provide the program with an error status and allow execution to continue. The following example shows the syntax for this statement:

TRAP	{	<i>file - name</i>	}	[ON]
		<i>access - name</i>			OFF	
		<i>interface - name</i>				

Example

```
TRAP RECORD ON
```

TRUE function

The TRUE function returns the value TRUE (1). The following example shows the syntax for this function:

TRUE

Example

```
ERROR=TRUE
```

TXT function

The TXT function returns the text value of an arithmetic expression in numeric display format. The following example shows the syntax for this function:

TXT(*a*)

Example

```
TXT(100) returns "100"
```

UNPAD statement

The UNPAD statement removes padding characters from a text variable, array element, or substring. The following example shows the syntax for this statement:

UNPAD <i>string</i> [<i>padding</i>]	BEFORE
	AFTER
	ALL

Example

```
UNPAD CLIENT_NAME ALL
```

UNTIL-END statements

The UNTIL-END statements execute a block of statements repeatedly until a specified condition becomes true. MANTIS executes the block of statements in the range of the UNTIL-END once before it tests the condition. The following example shows the syntax for these statements:

UNTIL *expression*

```
.  
statements  
.  
END
```

Example

```
UNTIL YEAR=2000  
.  
.  
END
```

UPDATE statement

The UPDATE statement updates the contents of a record in a MANTIS file or in an external file. You need not read a record from a file before updating it. The following example shows the syntax for this statement:

UPDATE *file-name* [LEVEL=*level-number*]

Example

```
UPDATE RECORD
```

UPPERCASE function

The UPPERCASE function returns the text value of a text expression, changing any lowercase letters to uppercase. The following example shows the syntax for this function:

UPPERCASE(*t*)

Example

```
UPPERCASE("Lend Me Your Ears") returns "LEND ME YOUR EARS"
```

USER function

The USER function returns a text value containing the user name specified during MANTIS sign-on. The following example shows the syntax for this function:

USER

Example

```
IF USER="PAYROLL"
```

USERWORDS function

The USERWORDS function returns the number of MANTIS symbolic names currently in use. The following example shows the syntax for this function:

USERWORDS

Example

```
SHOW USERWORDS
```

VALUE function

The VALUE function returns the numeric value of a text expression. The following example shows the syntax for this function:

VALUE(*t*)

Example

```
VALUE("100") returns 100
```

WAIT statement

The WAIT statement suspends program execution until you press ENTER or any other action key. The following example shows the syntax for this statement:

WAIT

Example

```
WAIT
```

WHEN-END statements

The WHEN-END statements execute a block of statements when a specified condition is met. The following example shows the syntax for these statements:

WHEN *expression*

**.
statements**

**.
[WHEN *expression*
.
statements
.]**

END

Example

```
WHEN MAP= "PF1 "  
.  
.  
.  
END
```

WHILE-END statements

The WHILE-END statements execute a block of statements repeatedly as long as a specified condition is TRUE. The following example shows the syntax for these statements:

WHILE *expression*

**.
statements
.
END**

Example

```
WHILE RECORD<>"END"  
.  
.  
.  
END
```

ZERO function

The ZERO function returns the value zero. The following example shows the syntax for this function:

ZERO

Example

```
COUNTER=ZERO
```

A

Special characters

This section describes the special characters that are used with MANTIS for Windows. For more information about special characters and character sets, refer to the *MANTIS for Windows Language Reference Manual*, P19-2302 or the *MANTIS for Windows Facilities Reference Manual*, P19-2301. The following table describes the special characters:

Character	Meaning
"	Double quotes enclose a text literal.
'	A single quote (apostrophe) marks a continuation line.
()	Parentheses enclose subscripts, subexpressions, arguments, and parameters.
:	A colon separates two program statements on the same line. A colon also abbreviates the SHOW command, as described in <i>MANTIS for Windows Facilities Reference Manual</i> , P19-2301.
;	A semicolon inserts a space in a line displayed by the SHOW statement.
,	A comma separates subscripts, arguments, and parameters. It also inserts spaces up to the next zone in a line displayed by the SHOW statement.
.	A period designates the decimal point in a number.
_	An underline may be used in a symbolic name.
!	A broken vertical bar precedes a comment in a program line.
+	A plus sign adds one number to another or joins one text value to another.
-	A minus sign subtracts one number from another or reduces one text value by another.
*	An asterisk multiplies one number by another.
**	Two asterisks raises one number to the power of another.

Character	Meaning
/	A slash divides one number by another.
=	An equals sign tests whether one value is equal to another, or assigns a value to a variable.
<	A less-than sign tests whether one value is less than another.
>	A greater-than sign tests whether one value is greater than another.
< >	A less-than sign followed by a greater-than sign tests whether one value is unequal to another.
>=	A greater-than sign followed by an equals sign tests whether one value is greater than or equal to another.
<=	A less-than sign followed by an equals sign tests whether one value is less than or equal to another.
&	An ampersand indicates an indirect reference to a symbolic name.
@	An at sign directs MANTIS to obtain keyboard input from a DOS file (programming mode only).
\$	A dollar sign is short for the PERFORM command. (The dollar sign can be used as a command only, not in a programming statement.)

B

Attributes

This section lists the field-level, map-level, and device-level attributes that can be set. Underlining indicates the minimum number of characters you can specify for a particular attribute (e.g., AUT for AUTOSKIP). Field-level and map-level attributes are described in the *MANTIS for Windows Facilities Reference Manual*, P19-2301. Device-level attributes are discussed in the *MANTIS for Windows Administration Guide*, P19-2304.

Field-level attributes

<u>AUTOSKIP</u>	<u>NO AUTOSKIP</u>	
<u>BLINK</u>	<u>NO BLINK</u>	
<u>BRIGHT</u>	<u>NORMAL</u>	<u>HIDDEN</u>
<u>CURSOR</u>		
<u>DETECTABLE*</u>	<u>NON DETECTABLE*</u>	
<u>HIGHLIGHT</u>	<u>NO HIGHLIGHT</u>	
<u>LEFT BAR*</u>	<u>NO LEFT BAR*</u>	
<u>MODIFIED*</u>	<u>UNMODIFIED*</u>	
<u>NO COLOR</u>	<u>BLUE</u> , <u>GREEN</u> , <u>NEUTRAL</u> , <u>PINK</u> , <u>RED</u> , <u>TURQUOISE</u> , <u>YELLOW</u>	
<u>OVERLINE*</u>	<u>NO OVERLINE*</u>	
<u>PROTECTED</u>	<u>UNPROTECTED</u>	
<u>RESET</u>		

* This attribute has no effect in MANTIS for Windows. It is included for compatibility with MANTIS for the IBM mainframe.

<u>REVERSE</u>	<u>VIDEO</u>
<u>VIDEO</u>	
<u>RIGHT BAR*</u>	<u>NO RIGHT BAR*</u>
<u>UNDERLINE</u>	<u>NO UNDERLINE</u>
<u>UPPERCASE**</u>	<u>LOWERCASE**</u>

* This attribute has no effect in MANTIS for Windows. It is included for compatibility with MANTIS for the IBM mainframe.

** The UPPERCASE and LOWERCASE attributes have an effect in MANTIS for Windows, but are not supported by MANTIS for the IBM mainframe.

Map-level attributes

ALARM

PROTECTED BOTTOM LINE

FULL SCREEN DISPLAY

WINDOWING

NO ALARM

BOTTOM LINE

UNPROTECTED

NOT FULL SCREEN DISPLAY

NO WINDOWING

Device-level attributes

<u>SPOOL ON CLOSE</u>	<u>NO SPOOL ON CLOSE</u>
<u>CLASS</u>	
<u>BLINK</u>	<u>NO BLINK</u>
<u>BRIGHT</u>	<u>NORMAL</u>
<u>COLOR</u>	<u>NO COLOR</u>
<u>DETECTABLE</u>	<u>NON DETECTABLE</u>
<u>REVERSE VIDEO</u>	<u>VIDEO</u>
<u>UNDERLINE</u>	<u>NO UNDERLINE</u>

C

Operators

The following table describes some of the operators used by MANTIS for Windows. Refer to the *MANTIS for Windows Language Reference Manual*, P19-2302, for more information about operators.

&	Indirect reference
Unary + -	Unary signs (negation)
**	Exponentiation
* /	Multiplication and division
+ -	Addition and subtraction
=, >, <, >=, <=, <>	Relational tests
AND	Logical conjunction
OR	Logical disjunction

Index

\$

\$SYMBOL function 62

A

ABS function 33
ACCESS statement 33
access-name function 34
ALTER command 23, 24
at sign (@) 72
ATN function 34
ATTRIBUTE statement 34
ATTRIBUTE(PRINTER) function
35

B

BIG statement 35
BIND command 23
BREAK statement 35

C

CALL statement 36
CHAIN statement 36
CHANGE command 24
CHR function 36
CLEAR statement 37
commands
 ALTER 23
 BIND 23
 CHANGE 24
 CLEAR 24
 COPY 24
 DISPLAY 25
 DOWN 25
 EDIT 25
 ERASE 26
 GO 26
 HELP 26
 LIST 27
 LOAD 27

NEW 27
PURGE 28
QUIT 28
REPLACE 28
RUN 29
SAVE 29
SEQUENCE 29
SET 30
SHOW 30
UP 30
USAGE 31
COMMIT statement 37
CONVERSE statement 38
COPY command 24
COS function 38
CURSOR function 39

D

DATAFREE function 39
DATE function 39
DATE statement 39
DELETE statement 40
DEQUEUE statement 40
DISPLAY command 25
DO statement 40
DOLEVEL function 41
dollar sign (\$) 72
DOWN command 25

E

E function 41
EDIT command 25
EDIT statement 41
ENQUEUE statement 42
ENTRY-EXIT statement 42
ERASE command 26
EXP function 43

F

FOR-END statement 43
FORMAT function 44
FSI function 44
function
 \$SYMBOL 62
 ABS 33
 access-name 34
 ATN 34
 ATTRIBUTE(PRINTER) 35
 CHR 36
 COS 38
 CURSOR 39
 DATAFREE 39
 DATE 39
 DOLEVEL 41
 E 41
 EXP 43
 FORMAT 44
 FSI 44
 INT 46
 interface-name 47
 KEY 47
 LANGUAGE 47
 LOG 48
 MODIFIED 48
 NOT 49
 NULL 49
 NUMERIC 50
 PASSWORD 51
 PI 52
 POINT 52
 PRINTER 52
 PROGFREE 53
 RELEASE 54
 RND 55
 screen-name 56
 SGN 57
 SIN 58
 SIZE 59
 SQR 62
 TAN 63
 TERMINAL 63
 TERMSIZE 63
 TIME 64
 TRUE 65
 TXT 65
 UPPERCASE 67
 USER 67
 USERWORDS 68
 VALUE 68

ZERO 70

G

GET statement 44
GO command 26

H

HEAD statement 45
HELP command 26

I

IF-ELSE-END statement 45
indirect reference 72
INSERT statement 46
INT function 46
INTERFACE statement 46
interface-name function 47

K

KEY function 47

L

LANGUAGE function 47
LET statement 48
LIST command 27
LOAD command 27
LOG function 48

M

MODIFIED function 48

N

NEW command 27
NEXT statement 49
NOT function 49
NULL function 49
NUMERIC function 50

O

OBTAIN statement 50
 OUTPUT statement 50

P

PAD statement 51
 PASSWORD function 51
 PERFORM statement 51
 PI function 52
 POINT function 52
 PRINTER function 52
 PRINTER= statement 53
 PROGFREE function 53
 PROGRAM statement 53
 PROMPT statement 54
 PURGE command 28

Q

QUIT command 28

R

RELEASE function 54
 RELEASE statement 54
 REPLACE command 28
 RESET statement 55
 RETURN statement 55
 RND function 55
 RUN command 29

S

SAVE command 29
 SCREEN statement 56
screen-name function 56
 SCROLL statement 56
 SEED statement 57
 SEQUENCE command 29
 SET command 30
 SET statement 57
 SGN function 57
 SHOW command 30
 SHOW statement 58
 SIN function 58
 SIZE function 59
 SLICE statement 59
 SLOT statement 60
 SMALL statement 60, 61

SQR function 62
 statement
 ACCESS 33
 ATTRIBUTE 34
 BIG 35
 BREAK 35
 CALL 36
 CHAIN 36
 CLEAR 37
 COMMIT 37
 CONVERSE 38
 DATE 39
 DELETE 40
 DEQUEUE 40
 DO 40
 EDIT 41
 ENQUEUE 42
 ENTRY-EXIT 42
 FOR-END 43
 GET 44
 HEAD 45
 IF-ELSE-END 45
 INSERT 46
 INTERFACE 46
 LET 48
 NEXT 49
 OBTAIN 50
 OUTPUT 50
 PAD 51
 PERFORM 51
 PRINTER= 53
 PROGRAM 53
 PROMPT 54
 RELEASE 54
 RESET 55
 RETURN 55
 SCREEN 56
 SCROLL 56
 SEED 57
 SET 57
 SHOW 58
 SLICE 59
 SLOT 60
 SMALL 60, 61
 STOP 62
 TEXT 64
 TIME 64

statement (*cont.*)

- TRAP 65
- UNPAD 66
- UNTIL-END 66
- UPDATE 67
- WAIT 68
- WHEN-END 69
- WHILE-END 70
- STOP statement 62

T

- TAN function 63
- TERMINAL function 63
- TERMSIZE function 63
- TEXT statement 64
- TIME function 64
- TIME statement 64
- TRAP statement 65
- TRUE function 65
- TXT function 65

U

- UNPAD statement 66
- UNTIL-END statement 66
- UP command 30
- UPDATE statement 67
- UPPERCASE function 67
- USAGE command 31
- USER function 67
- USERWORDS function 68

V

- VALUE function 68

W

- WAIT statement 68
- WHEN-END statement 69
- WHILE-END statement 70

Z

- ZERO function 70

Reader Comment Sheet

Name: _____

Job title/function: _____

Company name: _____

Address: _____

Telephone number: _____ Date: _____

How often do you use this manual? ☐ Daily ☐ Weekly ☐ Monthly ☐ Less

How long have you been using this product? ☐ Months ☐ Years

Can you find the information you need? ☐ Yes ☐ No Please comment.

Is the information easy to understand? ☐ Yes ☐ No Please comment.

Is the information adequate to perform your task? ☐ Yes ☐ No Please comment.

General comment: _____

WE STRIVE FOR QUALITY

To respond, please fax to Larry Fasse at (513) 612-2000.